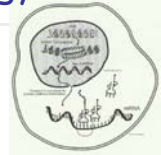


## Supervised learning and analysis of microarray data

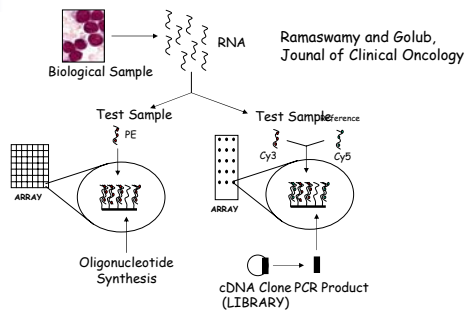
Devika Subramanian  
Comp 470

## Microarray technology

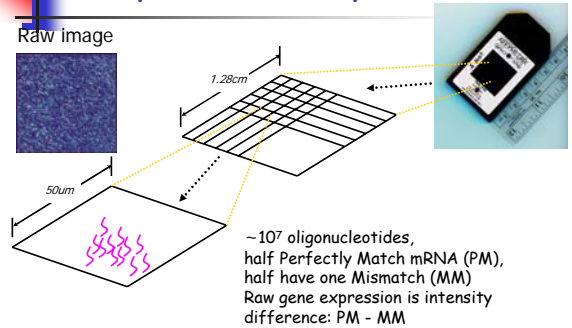
- Quick recap
  - Proteins: state of cell
  - Gene: codes for a protein
  - mRNA: helps assemble a protein
  - mRNA levels ~ gene exp. level ~ protein levels
- Microarrays measure the expression levels of thousands of genes at a time.
- Typical experiment: Measure expression of genes under different conditions and ask what is different at a molecular level and why.



## Microarrays



## Affymetrix arrays



## Microarray applications

- Biological discovery
  - new and better molecular diagnostics
  - new molecular targets for therapy
  - finding and refining biological pathways
- Recent examples
  - molecular diagnosis of leukemia, breast cancer.
  - appropriate treatment for genetic signature
  - potential new drug targets

## Two computational tasks

- Classifying gene expressions: **this week**
  - What can be learnt about a cell from the set of all mRNA expressed in a cell? Classifying diseases: does a patient have benign prostate cancer or metastatic prostate cancer?
- Inferring regulatory networks: **next week**
  - What is the "circuitry" of the cell? What are the genetic pathways of cancer?

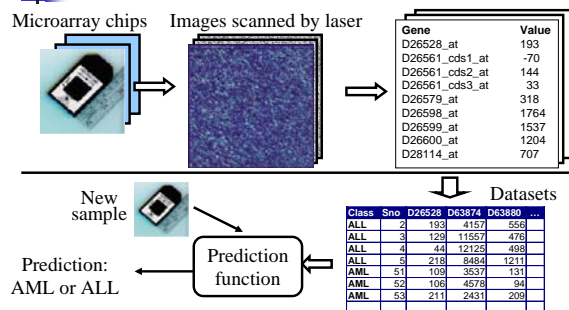
## Common Approaches

- Comparing two measurements at a time
  - Person 1, gene  $G$ : 1000
  - Person 2, gene  $G$ : 3200
  - Greater than 3-fold change: flag this gene
- Comparing one measurement with a population of measurements... is it unlikely that the new measurement was drawn from same distribution?

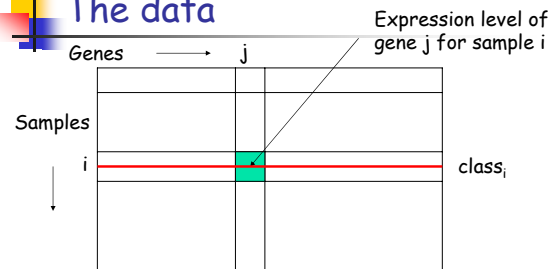
## Classification

- Use our knowledge of class values, e.g., myeloma vs. normal etc., to gain added insight.
- Find genes that are best predictors of class.
  - Can provide useful tests, e.g. for choosing treatment.
  - If predictor is **comprehensible**, may provide novel **insight**, e.g., point to a new therapeutic target.

## Classifying gene exp data

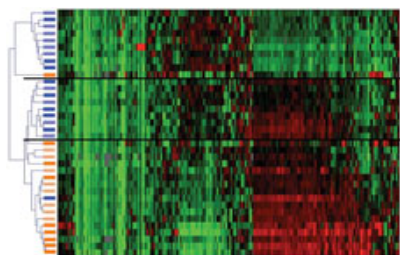


## The data



Samples are labeled.  
Red line denotes vector of exp levels for sample  $i$

## Heat maps



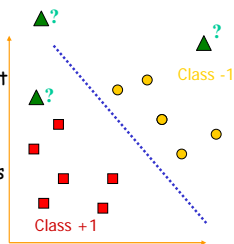
## Challenges

- Microarray data inherit large experimental and biological variances
  - experimental bias + tissue heterogeneity
  - cross-hybridisation
  - 'bad design': confounding effects
- Microarray data are **sparse**
  - high-dimensionality of genes
  - low number of samples/arrays
  - Curse of dimensionality*
- Microarray data are highly **redundant**
  - Many genes are co-expressed, thus their expression is strongly correlated.

## Classification

Given examples drawn from two classes, learn to classify new examples into the correct class.

Each point represents a vector of gene expression levels

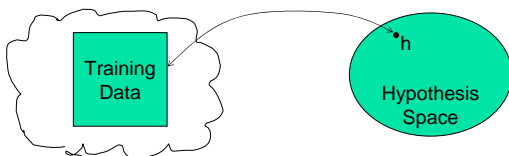


## The classification problem

- Given training data  $\{(x_1, y_1), \dots, (x_m, y_m)\}$ ,  $x_i$  in  $\mathbb{R}^n$ ,  $y_i$  in  $\{+1, -1\}$ .
- Estimate function  $h: \mathbb{R}^n \rightarrow \{+1, -1\}$  such that  $h$  will correctly classify **new** unseen examples from the same underlying probability distribution as the training data.

## Classification as optimization

- Set  $S$  of training data points
- Class  $H$  of hypotheses/models
- Optimization problem: Find the hypothesis/model  $h$  in  $H$  that best fits all data.



## Objective function

- Minimizing training set error does not imply minimizing true error!

$$R_{\text{train}}[h] = \frac{1}{m} \sum_{i=1}^m \frac{1}{2} [h(x_i) - y_i]^2 \quad \text{Empirical risk}$$

$$R[h] = \int \frac{1}{2} [h(x_i) - y_i]^2 dP(x, y) \quad \text{True error}$$

## Statistical machine learning theory

- Non-asymptotic theory, based on finite samples which bounds true error in terms of training set error.
- Gives tradeoff between complexity of model and amount of data needed to learn it.

## A bound on true error

- VC dimension theory allows us to relate train and test error for particular function classes. The key intuition is that the error of a function is not an absolute, but relative to the class of functions it is drawn from.

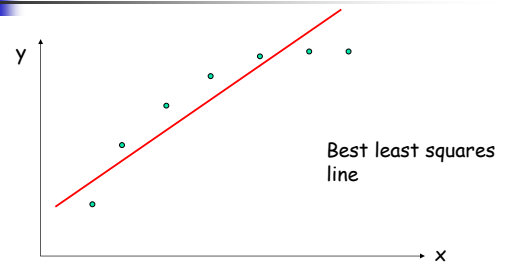
$$R[h] \leq R_{\text{train}}[h] + \sqrt{\frac{VC(h)(\log 2m / VC(h) + 1) - \log(\delta / 4)}{m}}$$

$VC(h)$  is the VC dimension of the class from which  $h$  is drawn and  $\delta$  is the probability bound,  $m$  is the size of the training set (Vapnik, 1995).

## Tradeoffs

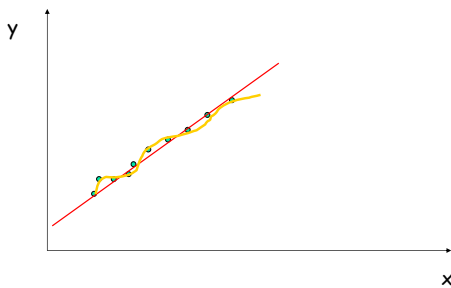
- With only a small amount of data, we can only discriminate between a small number of different hypotheses.
- As we get more data, we have more evidence, so we can consider more alternative hypotheses.
- Complex hypotheses give better fit to the data.

## Simple hypothesis will under-fit



Cannot take advantage of more data!

## Complex hypotheses will overfit



## Adaptive hypothesis space selection

- Find hypothesis  $h$  to minimize  $\text{error}(h) + \lambda \text{ complexity}(h)$

Regularization

## Support vector machines

- A new generation of learning algorithms based on
  - Non-linear optimization
  - Statistics
  - Functional analysis
- Come with theoretical guarantees on performance, because the learning problem can be reduced to convex optimization.

## Applications

- SVMs have been used in a wide variety of tasks and are reputed to be the best for
  - Text categorization
  - Handwriting recognition
  - Classification of gene expression data

## History

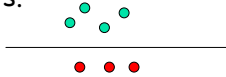
- Introduced in 1992 by Boser, Guyon and Vapnik (COLT 1992).
- Very rapid growth since then. 2 excellent textbooks and lots of new work both in theory and applications.
- [www.kernel-machines.org](http://www.kernel-machines.org) is a great resource for learning about SVMs.

## The Problem

- Given training data  $\{(x_1, y_1), \dots, (x_m, y_m)\}$ ,  $x_i$  in  $\mathbb{R}^n$ ,  $y_i$  in  $\{+1, -1\}$ .
- Estimate function  $h: \mathbb{R}^n \rightarrow \{+1, -1\}$  such that  $h$  will correctly classify new unseen examples from the same underlying probability distribution as the training data.

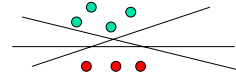
## Linear support vector machines

- Consider the class of oriented hyperplanes in  $\mathbb{R}^n$ .
  - $h(x) = \text{sign}(w \cdot x + b)$
- If data is linearly separable, then there is a function from this class that separates the +1 points from the -1 points.



## Linear separating hyperplanes

- Unfortunately, there are an infinite number of linear hyperplanes that separate the data!



## Geometric Margin

Coordinates of  $B = x - d \frac{w}{\|w\|}$

$B$  lies on line defined by  $w^T x + b = 0$

$$w^T \left( x - d \frac{w}{\|w\|} \right) + b = 0$$

Solving for  $d$ ,

$$d = \frac{w^T x + b}{\|w\|}$$

## Geometric interpretation

The optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes and intersects it halfway between them.

## Margin maximization

- Let  $x^+$  and  $x^-$  be the two points on the convex hulls of the positive and negative data which are closest to the maximal margin hyperplane.

$$1. w^T x^+ + b = +1$$

$$2. w^T x^- + b = -1$$

$$3. x^+ = x^- + \lambda \frac{w}{\|w\|}$$

Lambda is the margin width, It is inversely proportional to w.w. So to maximize margin, we minimize w.

$$w^T (x^+ - x^-) = 2, \text{ from 1. and 2.}$$

$$\lambda = \frac{2}{\|w\|}, \text{ from 3 and above.}$$

## Optimal separating hyperplane

- Among all separating hyperplanes, there is one with the maximum **margin**.
- A hyperplane separating data  $(x_1, y_1), \dots, (x_m, y_m)$  satisfies
  - $(w \cdot x_i) + b \geq 1$  if  $y_i = +1$
  - $(w \cdot x_i) + b \leq -1$  if  $y_i = -1$
- Or in short...
  - $y_i [(w \cdot x_i) + b] \geq 1$ , for  $i = 1..m$
- The optimal hyperplane satisfies the above conditions and has the minimal norm  $\|w\|^2 = w \cdot w$

## Learning the maximum margin classifier

Find  $w$  and  $b$  that minimize

$$\tau(w) = \frac{1}{2} \|w\|^2$$

subject to

$$y_i (w^T x_i + b) \geq 1, \text{ for } i = 1..m$$

Quadratic programming!

## Solving the quadratic program

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i (w^T x_i + b) - 1)$$

$L$  must be minimized with respect to  $w$  and  $b$  and maximized with respect to the Lagrange multipliers  $\alpha_i$ .

The first derivative with respect to  $w$  and  $b$  must vanish at the saddle point.

## Solving the quadratic program

$$\frac{\partial L(w, b, \alpha)}{\partial w} = 0 \text{ which yields } \sum_{i=1}^m \alpha_i y_i x_i = w$$

This means  $w$  has an expansion in terms of a subset of the training data, namely those  $(x_i, y_i)$  for which  $\alpha_i > 0$ . These data points are called **support vectors**. None of the other data points matter. The maximal margin hyperplane is completely determined by the support vectors.

## Solving the quadratic program

$$\frac{\partial L(w, b, \alpha)}{\partial b} = 0 \text{ which yields } \sum_{i=1}^m \alpha_i y_i = 0$$

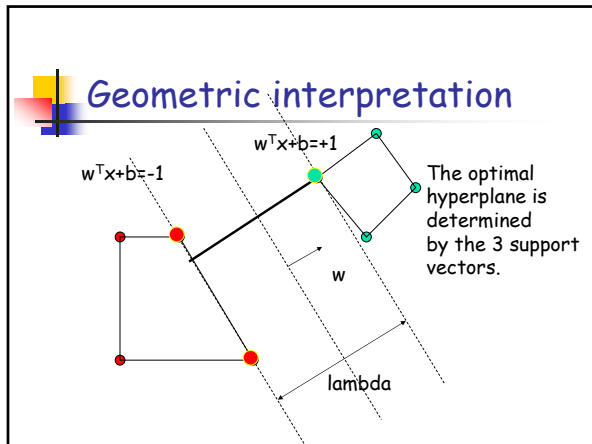
$$\alpha_i \geq 0, i = 1..m$$

$$y_i (w^T x_i + b) - 1 \geq 0, i = 1..m$$

By the KKT complementarity condition,

$$\alpha_i (y_i (w^T x_i + b) - 1) = 0, i = 1..m$$

Support vectors lie on the margin, because when  $\alpha_i > 0$ , then  $y_i (w \cdot x_i + b) - 1 = 0$ .



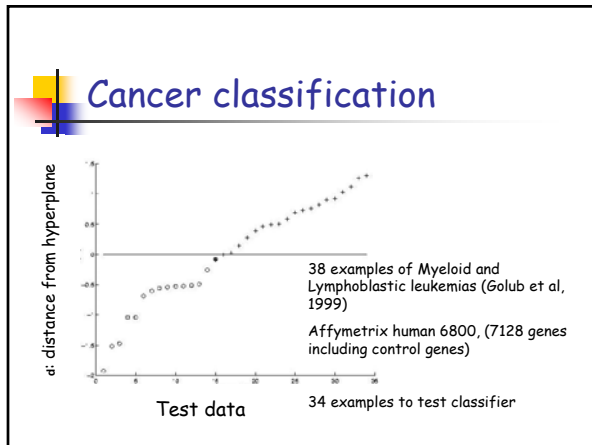
### Solution

$$h(x) = \text{sign}(w^T x + b)$$

$$= \text{sign}\left(\sum_{i=1}^m (y_i \alpha_i (x^T x_i) + b)\right)$$

The hyperplane decision function uses the **support vectors alone**, and takes the **dot product** of the support vectors with  $x$ .

Note:  $b$  is calculated from the KKT comp. condn.



### Extension to non-separable data

- Idea #1: soft margin hyperplane

Slack variables

### Soft margin hyperplanes

Minimize  $\frac{1}{2} \|w\|^2 + c \sum_i \xi_i^\delta, \delta \geq 0$

subject to

$$y_i (w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

For  $\delta = 1$ , this is a convex optimization problem. We can set up the Lagrangian and solve for  $w, b$  and  $\xi_i$  using the KKT conditions.

### Solving the opt. problem

$$L(w, b, \alpha, \xi) = \frac{1}{2} \|w\|^2 + c \sum_{i=1}^m \xi_i$$

$$- \sum_{i=1}^m \alpha_i (y_i (w \cdot x_i + b) - 1 + \xi_i)$$

$$- \sum_{i=1}^m \mu_i \xi_i$$

## The KKT conditions

$$\frac{\partial L(w, b, \alpha, \xi)}{\partial w} = 0 \text{ which yields } w = \sum_{i=1}^m \alpha_i y_i x_i$$

$$\frac{\partial L(w, b, \alpha, \xi)}{\partial b} = 0 \text{ which yields } \sum_{i=1}^m \alpha_i y_i = 0$$

$$\frac{\partial L(w, b, \alpha, \xi)}{\partial \xi_i} = 0 \text{ which yields } c - \alpha_i - \mu_i = 0$$

$$\frac{\partial L(w, b, \alpha, \xi)}{\partial \alpha_i} = 0 \text{ which yields } y_i (w^T x_i + b) - 1 + \xi_i \geq 0$$

KKT comp. condn.  $\alpha_i (y_i (w^T x_i + b) - 1 + \xi_i) = 0$

## The solution

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

From the KKT complementarity condition, we get support vectors are the training data points for which

$$y_i (w \cdot x_i + b) - 1 + \xi_i = 0$$

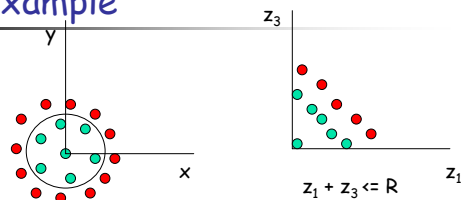
$$y_i (w \cdot x_i + b) = 1 - \xi_i$$

That is, support vectors lie on the margin!

## Non-linear support vector machines

- A generalization to handle the case when the decision function  $f$  is known to be not a linear function of the input  $x$ .
- Central idea: feature spaces. Map the  $x$  onto a higher dimensional feature space  $\phi(x)$ . Then, use linear support vector machines to obtain the optimal separating hyperplane in this high dimensional feature space.

## Example



$$\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi((x, y)) = (x^2, \sqrt{2}xy, y^2)$$

## Direct mapping

- Direct mapping to a high dimensional space suffers from the curse of dimensionality. To consider all  $d^{\text{th}}$  order products of an  $n$ -dimensional vector, we have to consider
  - $(n+d-1)! / (d!(n-1)!)$  terms
- For  $n = 16 \times 16$ ,  $d = 5$ , we have a  $10^{10}$  dimensional feature space.

## A closer look at decision fn

- Note that decision function is of the form

$$h(x) = \text{sign}(w^T x + b)$$

$$= \text{sign}\left(\sum_i \alpha_i y_i (x^T x_i) + b\right)$$

- We only use dot products of the input vectors for determining the optimal separating hyperplane.



## Kernels to the rescue

- If we want to find a separating hyperplane in the feature space, we need to compute the dot product of  $\phi(x)$  and  $\phi(x_i)$ .
- Define a kernel function  $K$  which returns the dot product of the images of its two arguments

$$K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

## Non-linear support vector machines

- The decision function is of the form

$$h(x) = \text{sign}(w^T \phi(x) + b)$$

$$= \text{sign}\left(\sum_i \alpha_i y_i (K(x, x_i)) + b\right)$$

- We only use dot products of the input vectors for determining the optimal separating hyperplane.

## Examples of kernels

- Polynomial kernel

$$K(x, y) = (x^T y)^d$$

- Second degree polynomial kernel

$$\phi((x_1, x_2)) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

$$\phi((y_1, y_2)) = (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$

$$K(x, y) = \phi(x)^T \phi(y) = (x_1^2 y_1^2 + 2x_1 x_2 y_1 y_2 + x_2^2 y_2^2)$$

$$= (x_1 y_1 + x_2 y_2)^2 = ((x_1, x_2)^T (y_1, y_2))^2 = (x^T y)^2$$

- Generalized polynomial kernel

$$K(x, y) = (x^T y + c)^d$$

## More kernels

- Exponential kernel (Gaussian RBF)

$$K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}$$

- Tanh kernel

$$K(x, y) = \tanh(kx^T y - \delta)$$

## Wolfe dual form

$$\text{Maximize } W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

subject to  $\alpha_i \geq 0; i = 1..m$

$$\sum_i \alpha_i y_i = 0$$

Derived by substituting for  $w$  and  $b$  into  $L(w, b, \alpha)$ .

Advantage: maximization expressed in terms of dot products of the  $x$ 's. Used for learning non-linear SVMs

## Mercer condition

- Identifies the class of functions for which  $K(x, y)$  is the dot product of  $\phi(x)$  and  $\phi(y)$ .
- See the excellent tutorial by C. Burges (available from [www.kernel-machines.org](http://www.kernel-machines.org)) for a discussion of this condition.

## General support vector machines

- We will substitute  $\phi(x)$  for  $x$  in our previous formulation.
- Solutions are of the form:

$$\begin{aligned}h(x) &= \text{sign}(w^T x + b) \\ &= \text{sign}\left(\sum_{i=1}^m \alpha_i y_i (\phi(x_i)^T \phi(x) + b)\right) \\ &= \text{sign}\left(\sum_{i=1}^m \alpha_i y_i K(x_i, x) + b\right)\end{aligned}$$

## SVM demo

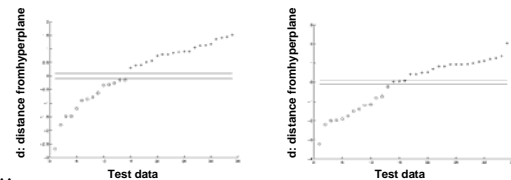
[Click here](#)

## Feature selection

- SVMs as stated use all genes/features.
- Molecular biologists/oncologists seem to be convinced that only a small subset of genes are responsible for particular biological properties, so they want the "relevant" genes.

## Results with feature selection

AML vs ALL: 40 genes 34/34 correct, 0 rejects.  
5 genes 31/31 correct, 3 rejects of which 1 is an error.



Mukherjee et. al. 2005

## Two feature selection techniques

- Recursive feature elimination (RFE): based upon perturbation analysis, eliminate genes that perturb the margin the least.
- Optimize leave one out (LOO): based on the optimized leave-one-out error of an SVM.

## Recursive feature elimination

1. Solve the SVM problem for vector  $w$
2. Rank order elements of vector  $w$  by absolute value
3. Discard input features/genes corresponding to those vector elements with small absolute magnitude (for smallest 10%)
4. Retrain SVM on reduced gene set and goto step (2)



## Leave one out estimator

- Leave one point out, train on the others, test on the left out point.
- Repeat this for every point in the training data.
- Leave-one-out estimate is almost unbiased.



## Leave-one-out feature selection

- Use the LOO estimator as an objective function in the search for subsets of features.